Main.ts
Main.ts

install load listener

load → hndLoad

click start Button → hideScreen1

click plants&dung&pesticide → buyProduct()

click plants&dung&pesticide → getTarget()

click fields → fieldClick

```
export let capital: number
export let maxPrice: number
export let ...Counter: number
export let clickedVegetable: string
```

hndLoad

_event: Event

let start: select start button

install click listener on start-button
call hideScreen1

select images in market
let carrot: getImageByID

represents all Products

install click listener for all products to
call buyProduct function

select images in storage
let carrot1: getImageByID
...

represents all Plants

install click listener for plants
call getTarget()

set global counters to zero
capital = 0;
...Counter = 0

hideScreen1

_event: Event

let screen1
let screen2

werden getauscht mit style.display

createFields wird aufgerufen

createFields

for loop with index 7 to iterate through rows

for loop with index 9 to create fields

install click listener for fields to
call fieldClick() function

call function readformData

_event: MouseEvent

readData

let formdata: select FormData

for of schleife to iterate through FormData.entries

[Price] [Startcapital]

priceVariation = number

capital = number

show capital entry in div

show storage counters which are still at zero

call changePrices function

setInterval(changePrices, 10000)

let carrotPrice — represents all Products

let carrotSellPrice — represents all Plants

changePrices

carrotPrice = random number * maxprice +1

carrotSellPrice = random number * maxprice +1 +2

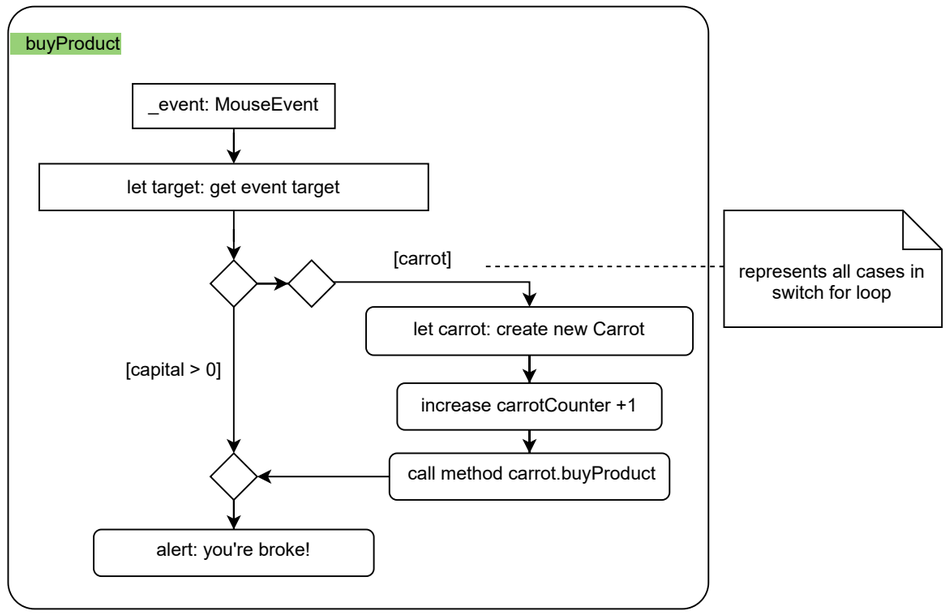
show prices and sellPrices in HTMLTableCellElement

getTarget

_event: MouseEvent

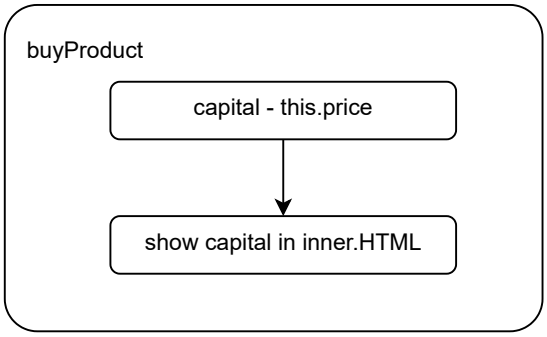
let target: event target

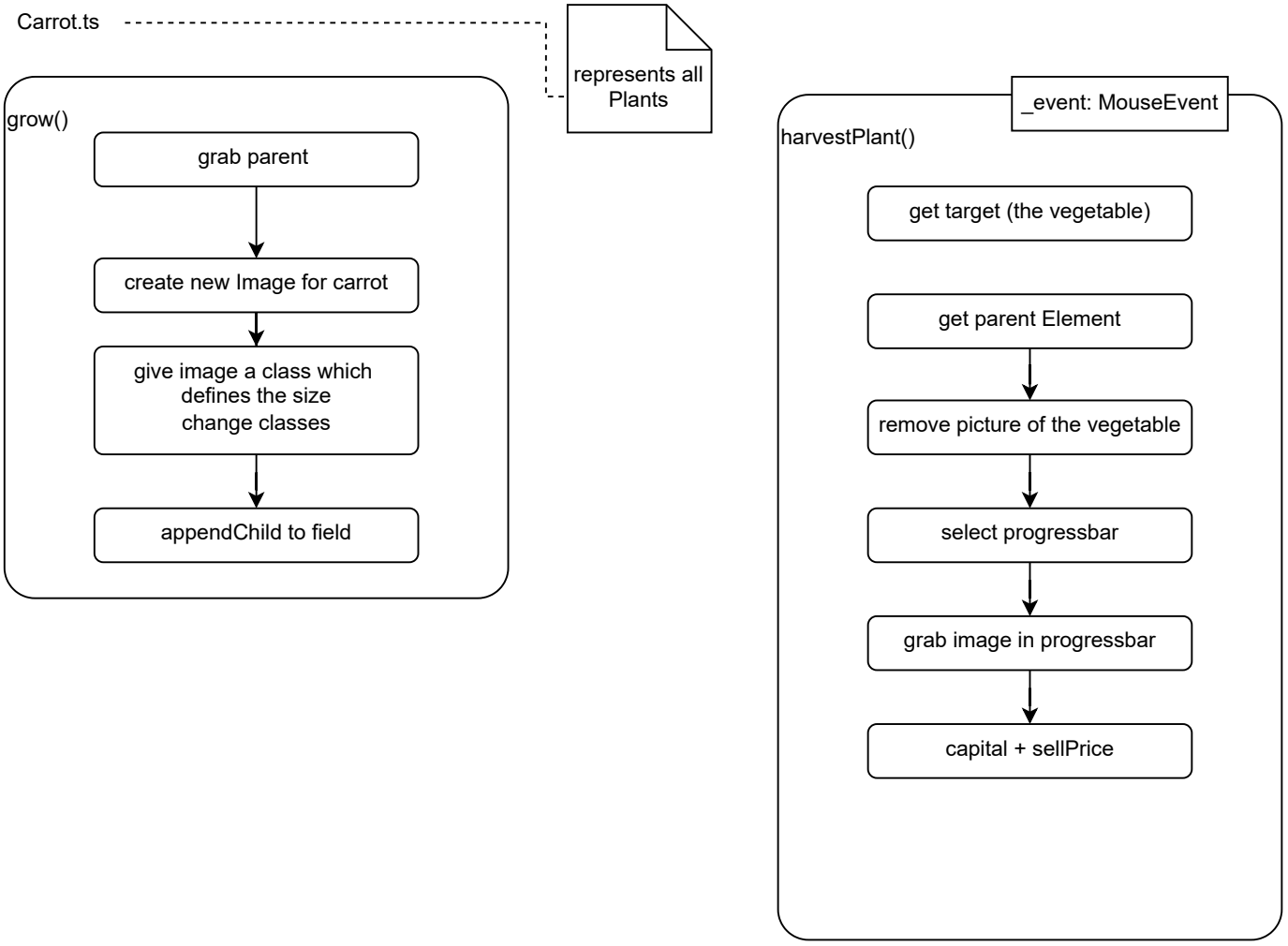
clickedVegetable = target.id


buyProduct
_event: MouseEvent
let target: get event target
[carrot]
represents all cases in switch for loop
let carrot: create new Carrot
[capital > 0]
increase carrotCounter +1
call method carrot.buyProduct
alert: you're broke!

fieldClick
_event: MouseEvent
_event: MouseEvent
let target: get event target
represents all cases in switch for loop
[carrot1]
break
[carrotCounter > 0]
let carrot: create new Carrot
carrot.grow()
decrease carrotCounter +1

Product.ts




Carrot.ts
represents all Plants
grow()
grab parent
create new Image for carrot
give image a class which defines the size
change classes
appendChild to field
_event: MouseEvent
harvestPlant()
get target (the vegetable)
get parent Element
remove picture of the vegetable
select progressbar
grab image in progressbar
capital + sellPrice